PyTorch for Mechanical Engineering

Table of Contents

Introduction to PyTorch

PyTorch is an open-source deep learning framework developed by Facebook. It is widely used for machine learning, neural networks, and AI applications, particularly in research and industrial settings. In mechanical engineering, PyTorch is used for predictive maintenance, defect detection, and process optimization.

Applications of PyTorch in Mechanical Engineering

- **Predictive Maintenance:** Using AI to predict machinery failure based on sensor data.
- **Defect Detection:** Identifying manufacturing defects through image classification.
- **Process Optimization:** AI-driven insights to enhance efficiency in industrial operations.
- **Material Property Prediction:** Deep learning for predicting the behavior of materials under stress.
- **Automated Quality Inspection:** AI-based vision systems for quality control.

Running PyTorch in Google Colab

Google Colab provides a cloud-based platform to run PyTorch without local installation. To use PyTorch in Colab, simply import it as follows:

```
import torch

print(torch.__version__)
```

PyTorch for Defect Detection in Manufacturing

A common application of PyTorch in mechanical engineering is **defect detection**, where deep learning models analyze images of manufactured components to classify them as defective or non-defective.

Example: Defect Detection Using a Neural Network

```
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
# Simulated defect dataset (features: size, surface roughness, color intensity)
x_train = torch.tensor([[5, 0.2, 100], [6, 0.3, 120], [4, 0.1, 90], [7, 0.5, 130], [3, 0.05, 80]], dtype=torch.float32)
y_train = torch.tensor([0, 1, 0, 1, 0], dtype=torch.float32).view(-1, 1)  # 0: No defect, 1: Defective
# Define a simple neural network model
class DefectClassifier(nn.Module):
    def __init__(self):
        super(DefectClassifier, self).__init__()
        self.fc1 = nn.Linear(3, 8)
        self.fc2 = nn.Linear(8, 4)
        self.fc3 = nn.Linear(4, 1)
        self.sigmoid = nn.Sigmoid()
        def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.sigmoid(self.fc3(x))
        return x
```

```
# Initialize model, loss function, and optimizer
model = DefectClassifier()
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
# Train the model
for epoch in range(100):
    optimizer.zero_grad()
    output = model(x_train)
    loss = criterion(output, y_train)
    loss.backward()
    optimizer.step()
# Predict defect probability for a new manufactured component
new_data = torch.tensor([[5.5, 0.25, 110]], dtype=torch.float32)
prediction = model(new_data).item()
print("Defect Probability:", prediction)
```

Mechanical Context:

- **Inputs:** Features such as size, surface roughness, and color intensity of manufactured components.
- **Output:** Predicts whether the component is defective or not.
- **Application:** Used in automated quality control systems to reduce defective products in manufacturing.

Conclusion

PyTorch enables mechanical engineers to integrate AI and deep learning into real-world applications, improving efficiency, reducing maintenance costs, and optimizing industrial processes. By leveraging neural networks, engineers can detect defects, predict failures, and automate quality control.

Additionally, PyTorch can be easily run in **Google Colab**, making it accessible for mechanical engineers to implement AI-driven solutions without requiring high-end hardware.